

Deploiement de sdg market en en local pour le développement

Pré-requis

Installation des PHP 7.2

Le backend utilisant PHP 7. Il nous faut dans un premier temps installer PHP sur notre OS. Pour cela, il suffit de suivre [ce guide](#) jusqu'au step 4 inclus (nous n'avons pas besoin de configurer apache) en prenant soin de télécharger la version 7.2 de php au moment du téléchargement.

Editez le fichier `c:\php\php.ini` et cherchez la variable `memory_limit` puis remplacer sa valeur comme suit :

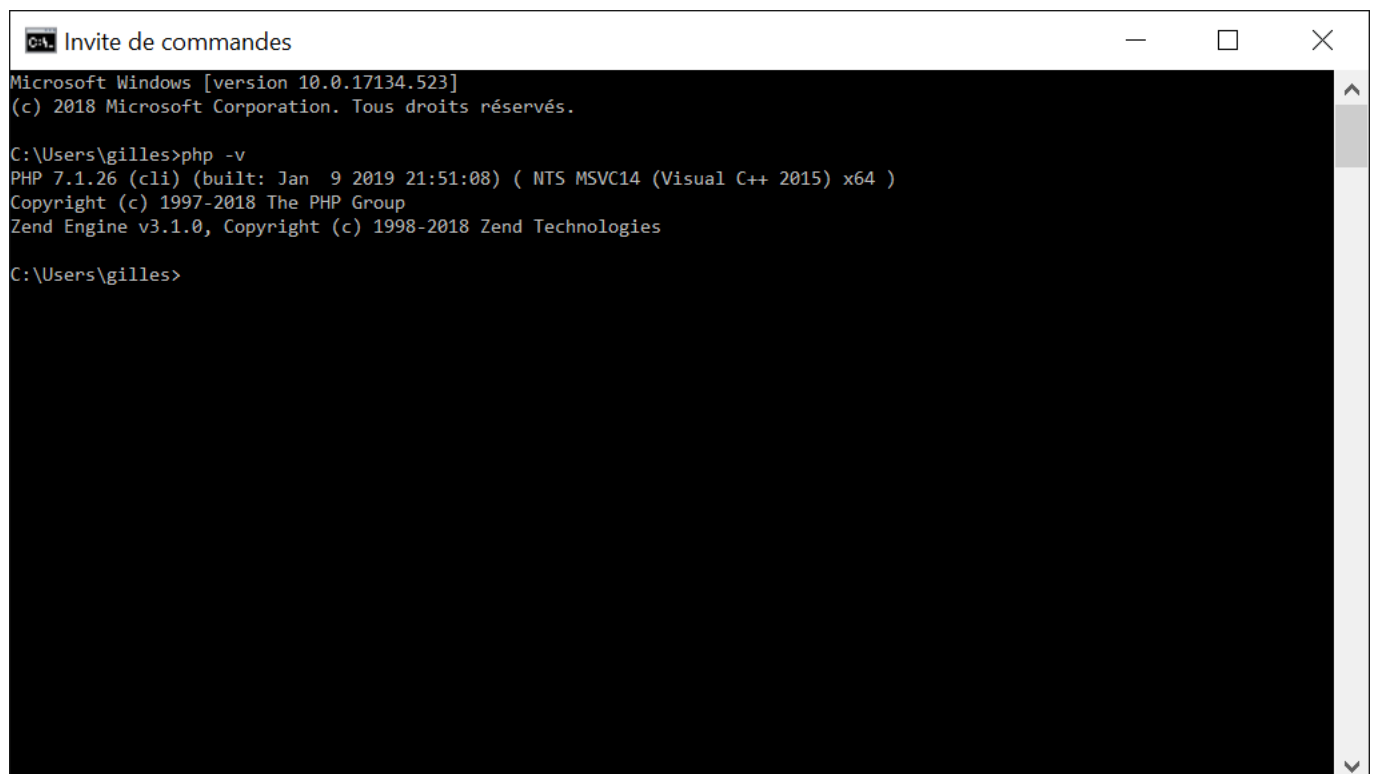
```
memory_limit = -1
```

Cette valeur supprime la limite de ram alouée à PHP. Ceci est utile car la librairie composer que nous allons voir tout à l'heure est très gourmande en mémoire vive.

Enfin, il faut télécharger le certificat suivant <http://curl.haxx.se/ca/cacert.pem> pour pouvoir utiliser curl et déplacez le dans `c:\php`

Editez le fichier `c:\php\php.ini` et

Une fois php installé, vous pouvez contrôler que votre installation fonctionne en ouvrant une fenêtre dos. Dans le menu démarrer, tapez "Invite de commande" et lancez l'application. Puis dans la fenêtre DOS, tapez `php -v` pour vérifier que tout fonctionne. Vous devriez obtenir le résultat suivant :



```
Microsoft Windows [version 10.0.17134.523]
(c) 2018 Microsoft Corporation. Tous droits réservés.

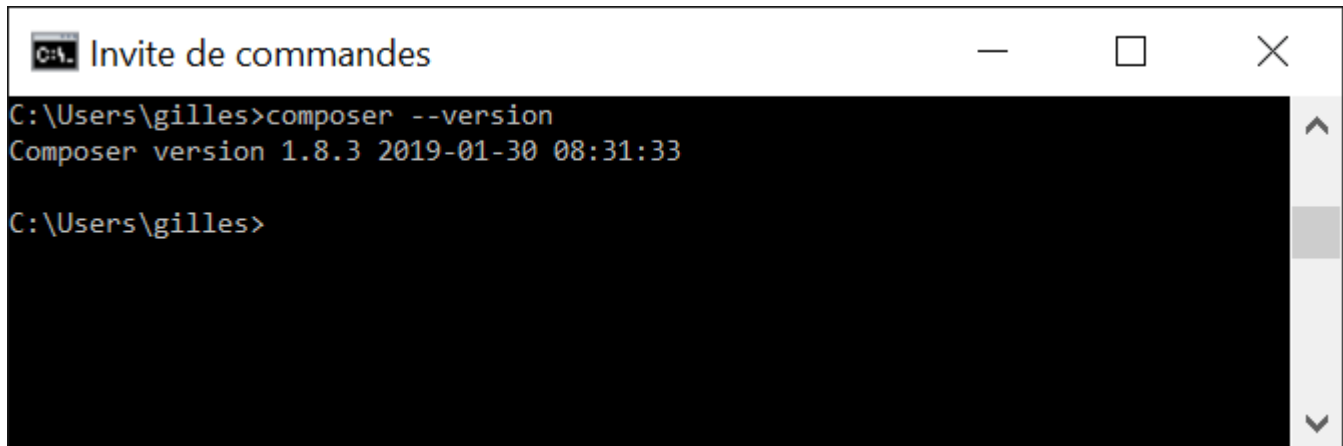
C:\Users\gilles>php -v
PHP 7.1.26 (cli) (built: Jan  9 2019 21:51:08) ( NTS MSVC14 (Visual C++ 2015) x64 )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.1.0, Copyright (c) 1998-2018 Zend Technologies

C:\Users\gilles>
```

Installation de composer

Composer est le gestionnaire de packages le plus répandu dans le monde PHP (équivalent de npm dans le monde de node). Pour initialiser le projet, il faut donc installer composer sur votre système. Pour télécharger l'application, rendez-vous sur le site de [composer](#) puis téléchargez et installez la version windows.

Une fois composer installé, vous pouvez contrôler que votre installation fonctionne en ouvrant une fenêtre dos. Dans le menu démarrer, tapez "Invite de commande" et lancez l'application. Puis dans la fenêtre DOS, tapez `composer --version` pour vérifier que tout fonctionne. Vous devriez obtenir le résultat suivant :



```
C:\Users\gilles>composer --version
Composer version 1.8.3 2019-01-30 08:31:33

C:\Users\gilles>
```

Installation de mysql

L'application utilisant une base de données MySQL, il faut installer MySQL ou MariaDB sur la machine. Vous pouvez installer MariaDB en suivant [ce guide](#). Téléchargez la version 10.3 (stable) [ici](#) avant de suivre le guide d'installation.

Maintenant vous pouvez aller dans le menu démarrer et lancer "Command prompt Maria DB" et taper :

```
mysql -uroot -p[YOUR_PASSWORD]

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.3.12-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database sdg_market;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> exit
Bye
```

Now you can use any mysql IDE (like mysql workbench) in order to create a database. Let's call it `sdg_market` for example.

Installation de node.js et npm

Le frontend utilisant des dépendances npm, nous avons besoin d'installer le gestionnaire de package npm pour pouvoir compiler le frontend de notre application.

Rendez-vous sur le site de node afin de télécharger et installer la version windows [à cette adresse](#). Prenez la version LTS (10.15.1).

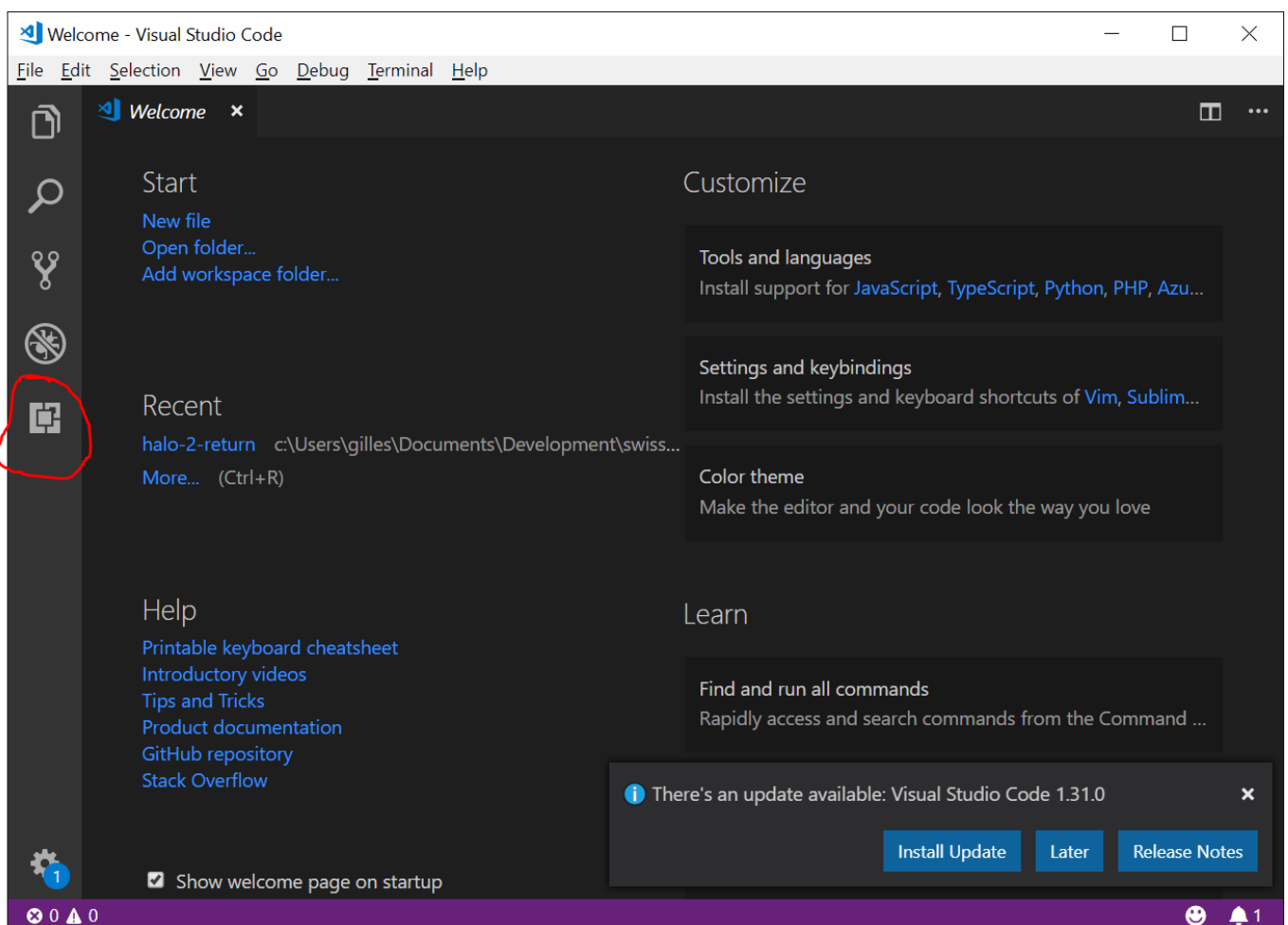
Une fois nodejs installé, vous pouvez contrôler que votre installation fonctionne en ouvrant une fenêtre dos. Dans le menu démarrer, tapez "Invite de commande" et lancez l'application. Puis dans la fenêtre DOS, tapez `npm --version` pour vérifier que tout fonctionne. Vous devriez avoir la version 6.4.1.

Installation de Visual Studio code

Visual studio code est un éditeur de code open source très complet (qui fonctionne un peu de la même manière que sublime text ou Atom). Nous allons baser notre guide sur l'utilisation de cet éditeur car il permet de tout faire, de la gestion du git à l'édition du code en passant par le café (presque).

Tout d'abord, téléchargez la dernière version de Visual studio code à cette [URL](#).

Ensuite il faudra installer des plugins pour être le plus efficient possible. Lancez Visual studio et cliquez sur CTRL+SHIFT+X ou cliquez sur l'icone ci-dessous afin d'ouvrir le gestionnaire d'extensions.



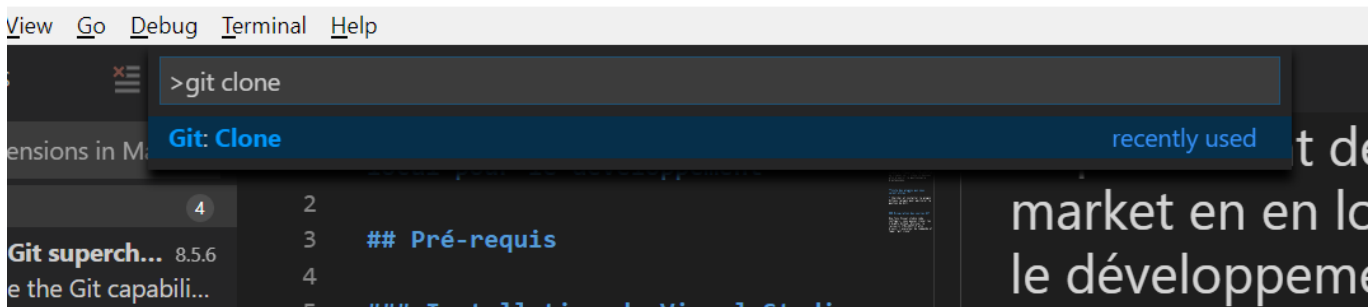
Visual studio code contient des tonnes de reccourcis très puissants, notamment la gestion de multi-curseurs qui, bien utilisés, peuvent faire gagner énormément de temps pendant la phase de développement, je vous conseil d'aller voir les différentes fonctionnalités proposées par cet IDE [ici](#)

Liste des plugins qui nous seront utiles pour commencer :

- Chercher et installer le plugin GitLens (plugin pour améliorer la gestion de Git).
- Chercher et installer le plugin "Git History diff" (très pratique pour afficher la date et le nom du dernier commit directement dans le code).
- Chercher et installer le plugin "PHP intellisense" (auto-completion PHP plus performante (à mon gout) que intellisense).
- Chercher et installer "PHP Getters & Setters" qui permettra de générer automatiquement les getters / setters après création de nouvelle propriétés dans les classes.
- Chercher et installer "phpfmt - PHP formatter (permet de formater automatiquement un fichier lors de la sauvegarde pour toujours respecter les normes PSR).
- Utiliser le raccourcis CTRL+SHIFT+P et chercher "Open Settings (JSON)" puis sélectionnez l'entrée. Une fois dedans, allez dans la partie "Users settings" et ajoutez `"editor.formatOnSave": true` dans le JSON.
- Cliquez sur "Reload" ou relancer le logiciel.

Récupération des sources GIT

2-return - Visual Studio Code



Une fois Visual studio code configuré, vous pouvez cloner les 2 dépôts Github. Utiliser le raccourci CTRL+SHIFT+P afin d'ouvrir l'exécuteur de commande et taper "git clone". Dans le champs, saisir l'adresse GIT de l'api (ici "https://github.com/sdgsolutionspace/sdg-market-api.git"). Puis sélectionnez l'emplacement où vous voulez enregistrer le code source. L'application vous demande si vous voulez ouvrir le dépôt. Cliquez sur "oui" afin d'ouvrir le projet. Procédez ensuite de la même manière pour le dépôt (https://github.com/sdgsolutionspace/sdg-market-frontend.git).

Création du workspace dans Visual studio code

Il est plus pratique de regrouper les 2 applications (frontend et backend) au sein d'un même projet pour éviter d'avoir plusieurs fenêtres ouvertes. Nous allons donc créer un workspace qui contient les 2 dossiers. Lorsque vous avez ouvert un des projets, faites simplement un clic droit dans la barre latérale gauche (en dessous des fichiers) et faites un clic droit "Add folder to workspace" puis sélectionner le 2ème dossier. Enfin, faites "File > Save workspace As" et sauvez votre espace de travail que vous pourrez ouvrir à n'importe quel moment pour avoir votre environnement tel que vous l'aviez configuré précédemment.

Démarrage du serveur PHP

Bon à savoir : Le backend est développé sur une base de Symfony 4.1 ([voir la doc ici](#)). Une mise à jour vers la 4.2 doit être planifiée car la 4.1 n'est plus maintenue. La partie API est gérée par le bundle FOSRestBundle ([voir la doc ici](#)).

Pour démarrer le serveur PHP, vous devez avoir compléter la partie "installation de PHP 7.2".

En premier lieu, il faut générer des certificats openssl pour la génération des tokens JWT. Comme il s'agit d'un environnement de dev, on peut utiliser un outil en ligne pour générer ces certificats pour s'épargner l'installation d'openssl. Rendez-vous sur [cette page](#) et cochez RSA et 4096 **sans spécifier de passphrase** puis :

- Enregistrer la clé publique dans un fichier config/jwt/public.pem
- Enregistrer la clé privée dans un fichier config/jwt/private.pem

Ouvrez une invite de commandes et naviguez jusqu'au dossier de votre application comme suit :

```
cd c:\Dossier\sdg-market-api
```

Editez le fichier `.env` et modifiez la ligne de configuration de la base de données avec les informations de la base de données que vous avez précédemment créé lors de l'installation de mariaDB.

```
DATABASE_URL=mysql://[YOUR_USER]:[YOUR_PASSWORD]@127.0.0.1:3306/[YOUR_DB_NAME]
```

Dans mon cas, j'ai :

```
DATABASE_URL=mysql://root:testtest@127.0.0.1:3306/sdg_market
```

Toujours dans le fichier `.env`, il faut maintenant configurer les clés API utilisées pour l'authentification des utilisateurs ainsi qu'un compte github qui sera utilisé pour parser les projets Git et générer les tokens.

Pour créer une app Github, il faut se connecter à github et se rendre sur la page <https://github.com/settings/apps> puis cliquer sur "New Github app". Ensuite il faut configurer l'app de la manière suivante :

7

Pull requests

Issues

Marketplace

Explore

Settings / Developer settings / GitHub Apps / git-trading

General

Permissions & events

Install App

Advanced

Public page

Basic information

GitHub App name

git-trading

The name of your GitHub App.

Description (optional)

This is displayed to users of your GitHub App.

Homepage URL

http://localhost:4200

The full URL to your GitHub App's website.

User authorization callback URL

http://localhost:4200/callback

The full URL to redirect to after a user authorizes an installation.

Setup URL (optional)

A URL to redirect users to after they install your GitHub App if additional setup is required on your end.

☐ Redirect on update

Redirect users to the Setup URL after installations are updated (E.g. repositories added/removed)

Webhook URL

http://localhost:4200/github/webhook

Events will POST to this URL. Read our [webhook documentation](#) for more information.

Webhook secret (optional)

Read our [webhook secret documentation](#) for more information.

Save changes

Cancel

Une fois l'application créée sur Github. Vous recevrez des clés de ce type :

OAuth credentials

- **Client ID:** lv1.54*****
- **Client secret:** d5f6d*****7831

Il n'y a plus qu'à les utiliser dans le fichier `.env` comme suit :

```
OAUTH_GITHUB_CLIENT_ID=<APP_CLIENT_ID> #ID reç
OAUTH_GITHUB_CLIENT_SECRET=<APP_CLIENT_SECRET>

GITHUB_PARSER_LOGIN=<YOUR_GITHUB_USERNAME>
GITHUB_PARSER_PASSWORD=<YOUR_GITHUB_PASSWORD>
```

Ensuite vous devez installer les dépendances composer avec la commande suivante :

```
composer install
php bin/console -v
Symfony 4.1.6 (kernel: src, env: dev, debug: true)
```

Si tout est ok, vous ne devez pas avoir d'erreur, et la version de symfony s'affiche, comme sur l'exemple ci-dessus.

Vous n'avez plus qu'à initialiser la base de données avec les commandes suivantes :

```
php bin/console doctrine:migrations:migrate
php bin/console cache:clear
```

Pour connaître l'ensemble des routes existantes dans le projet, une commande existe :

```
>php bin/console debug:router
```

Name	Method	Scheme	Host	Path
auth_github_auth_check	ANY	ANY	ANY	
/api/v1/connect/github/check				
api_login_check	ANY	ANY	ANY	
/api/login_check				
api_v1_exception_show_exception	PATCH	ANY	ANY	
/api/v1/exceptions/{exception}/show				
connect_check	GET	ANY	ANY	
/api/v1/checks/connect				
api_v1_git_project_get_git-projects	GET	ANY	ANY	
/api/v1/git-projects				
api_v1_git_project_get_git-project	GET	ANY	ANY	
/api/v1/git-projects/{gitProject}				
api_v1_git_project_post_git-project	POST	ANY	ANY	
/api/v1/git-projects				
api_v1_git_project_put_git-project	PUT	ANY	ANY	
/api/v1/git-projects/{gitProject}				
api_v1_git_project_delete_git-project	DELETE	ANY	ANY	
/api/v1/git-projects/{gitProject}				
api_v1_contributions_cget_contributions	GET	ANY	ANY	
/api/v1/contributions				
api_v1_contributions_get_contributions	GET	ANY	ANY	
/api/v1/contributions/{contribution}				
api_v1_contributions_post_contributions	POST	ANY	ANY	
/api/v1/contributions				
api_v1_contributions_put_contributions	PUT	ANY	ANY	
/api/v1/contributions/{contribution}				

api_v1_contributions_delete_contributions /api/v1/contributions/{contribution}	DELETE	ANY	ANY
api_v1_purchase_offer_get_purchase-offers /api/v1/purchase-offers	GET	ANY	ANY
api_v1_purchase_offer_get_purchase-offer /api/v1/purchase-offers/{purchaseOffer}	GET	ANY	ANY
api_v1_purchase_offer_post_purchase-offer /api/v1/purchase-offers	POST	ANY	ANY
api_v1_purchase_offer_put_purchase-offer /api/v1/purchase-offers/{purchaseOffer}	PUT	ANY	ANY
api_v1_purchase_offer_delete_purchase-offer /api/v1/purchase-offers/{purchaseOffer}	DELETE	ANY	ANY
api_v1_sell_offer_get_sell-offers /api/v1/sell-offers	GET	ANY	ANY
api_v1_sell_offer_get_sell-offer /api/v1/sell-offers/{sellOffer}	GET	ANY	ANY
api_v1_sell_offer_put_sell-offer_buy /api/v1/sell-offers/{sellOffer}/buy	PUT	ANY	ANY
api_v1_sell_offer_post_sell-offer /api/v1/sell-offers	POST	ANY	ANY
api_v1_sell_offer_put_sell-offer /api/v1/sell-offers/{sellOffer}	PUT	ANY	ANY
api_v1_sell_offer_delete_sell-offer /api/v1/sell-offers/{sellOffer}	DELETE	ANY	ANY
api_v1_transactions_get_transactions /api/v1/transactions	GET	ANY	ANY
api_v1_transactions_get_transaction /api/v1/transactions/{transaction}	GET	ANY	ANY
api_v1_transactions_post_transaction /api/v1/transactions	POST	ANY	ANY
api_v1_transactions_put_transaction /api/v1/transactions/{transaction}	PUT	ANY	ANY
api_v1_users_blacklist_user /api/v1/users/{user}/blacklist	PATCH	ANY	ANY
api_v1_users_assign_user_role /api/v1/users/{user}/roles/{role}/assign	PATCH	ANY	ANY
api_v1_users_user_user_token_refresh /api/v1/user/refresh-token	GET	ANY	ANY
api_v1_users_get_users /api/v1/users	GET	ANY	ANY
api_v1_users_get_user /api/v1/users/{username}	GET	ANY	ANY
api_v1_users_get_user_me /api/v1/user/me	GET	ANY	ANY
api_v1_users_post_user /api/v1/users	POST	ANY	ANY
app.swagger /api/doc.json	GET	ANY	ANY
_twig_error_test /_error/{code}.{_format}	ANY	ANY	ANY

Et enfin lancer un serveur de dev :

```
php bin/console server:run
```

Ouvrez votre navigateur sur l'url <http://localhost:8000/api/v1/contributions> afin de vous assurer que l'api fonctionne correctement. La page devrait renvoyer un tableau json vide [].

Une fois connecté avec un compte GitHub, vous pouvez vous donner les droits admin depuis l'invite de commande :

```
php bin/console app:promote <your_github_username>
```

Puis faites un logout / login sur la plateforme pour voir les options administrateurs.

Tips pour la partie backend

Ci-dessous les commandes utiles :

- `php bin/console` <- Cette commande permet d'afficher toutes les commandes disponible dans le CLI.
- `php bin/console --help` <- Pour n'importe quelle commande de la liste, le flag `--help` donne des informations sur l'utilisation de la commande et les arguments à utiliser
- `php bin/console app:promote <username>` <- Cette commande donne les droits admins à utilisateur.
- `php bin/console app:demote <username>` <- Cette commande supprime les droits admins à utilisateur.
- `php bin/console debug:router` <- Affiche l'ensemble des routes disponibles dans l'applications
- `php bin/console app:git-parse-and-assign` <- Parse les git configurés dans l'interface et génère les tokens pour les assigner aux utilisateurs.

Démarrage du serveur node pour servir le frontend en mode développement

A noter: Le frontend utilise angular 6. La documentation peut se trouver [ici](#).

Tout d'abord, ouvrez une invite de commande et naviguez jusqu'au dossier qui contient le code source du frontend :

```
cd c:\Dossier\sdg-market-frontend
```

Lancez l'installation des dépendences avec npm :

```
npm install
```

Et enfin, lancez le serveur dedéveloppement pour le frontend

```
npm run start
```

Si tout s'est bien passé, vous pouvez voir le frontend à l'adresse suivante : <http://localhost:4200>.